# – SAFAPS SIM –
# ARCHITECTURE DOCUMENT

Jeremy Harrault

SWORDFISH

## Objectives of this document

The purpose of this document is to present the architecture of the SAFAPS SIM project. It will contain diagrams as well as explanations to describe the architectural choices in order to fulfil the requirements. The information contained in the document act as a guide in order to fully develop, deploy and setup SAFAPS. Reflexions and reviewed decisions are tracked in this document.

## Glossary and Terminology

### – A –

API: Application Programming Interface

### – J –

JSON: JavaScript Object Notation

### – S –

S&F: Stress and Fatigue

SAFAPS: Stress and Fatigue Audit and Prediction Service

## Document Description

| | | |
|---|---|---|
| Title | SAFAPS SIM : Architecture Document | |
| Creation date | 25/01/2016 | |
| Publication date | 26/01/2016 | |
| Product Owner | Augustin Tataru | taau15md@student.ju.se |
| Authors | Jeremy Harrault | hajr15bp@ju.se |
| | | |
| Subject | Architecture Document | |
| Model version | 1.0 | |
| Document version | 0.6 | |

## Revisions table

| Date | Rev. | Author | Modified Section(s) | Comments |
|---|---|---|---|---|
| 25/01/16 | 0.1 | Jeremy Harrault | All | Add empty sections. |
| 28/01/16 | 0.2 | Jeremy Harrault | 4. | Add context and database view. |
| 29/01/16 | 0.3 | Jeremy Harrault | 4. | Add invoice table in database view and add additional information. |
| 02/02/16 | 0.4 | Jeremy Harrault | 1.<br>4. | Remove “Introduction and Management Summary” part.<br>Add the general architecture principals.<br>Add standardization from information viewpoint. |
| 04/02/16 | 0.5 | Jeremy Harrault | 4. | Explanation “datetime”.<br>Modify Database standardization rules |
| 05/02/16 | 0.6 | Jeremy Harrault | 4. | Move foreign key between Evaluation and Result into Evaluation table.<br>Add section for status of S&F evaluation<br>Add “status” column in database to “evaluation”. |

# Table of Contents



# List of Tables



# List of Figures

# 1. General Architecture Principles

## 1.1. Layer View

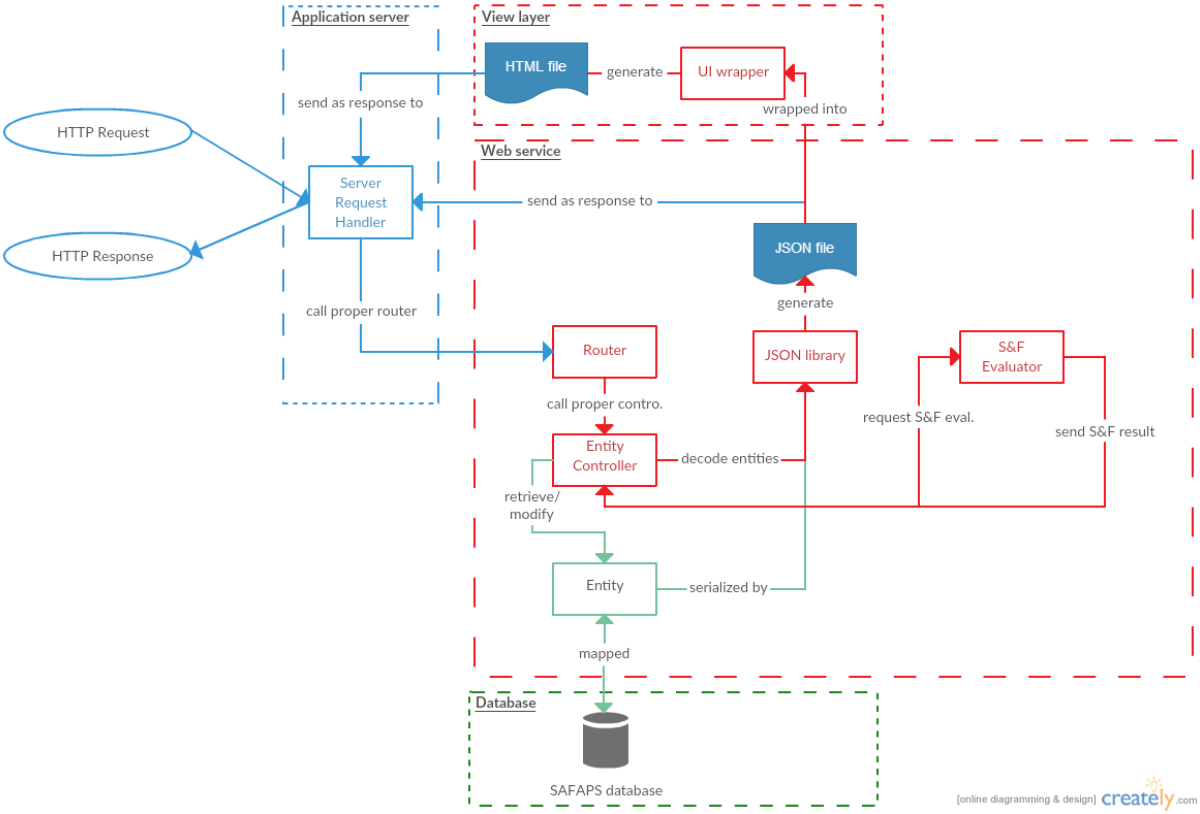


*Figure 1: SAFAPS SIM global architecture*

SAFAPS SIM Web server is able to receive HTTP requests and send HTTP response in return. It is composed of for layers.

| Layer | Description |
|---|---|
| Application server | The application server is first layer that the request goes across. This layer is between the software application and the operating system that the web server is running on. It catches the HTTP requests received by the server and pass it to the implemented web application. |
| Web service | The web service includes all business logic that will be computed. It includes the router which will call controllers depending on the resource pointed by the resource. Controllers can access and modify values within entities and call the S&F evaluator launching an asynchronous process. The web server takes JSON as input and returns JSON as output |
| Database | The layer is in charge of storing data for future use |
| View layer | The view layer is used to lay out the data returned by the web service within HTML to present data onto browser for example. |

*Table 1: SAFAPS SIM global layers*

## 1.2. Restful architecture

# 2. Architectural Design Decisions

# 3. Viewpoints and Views

## 3.1. Context Viewpoint

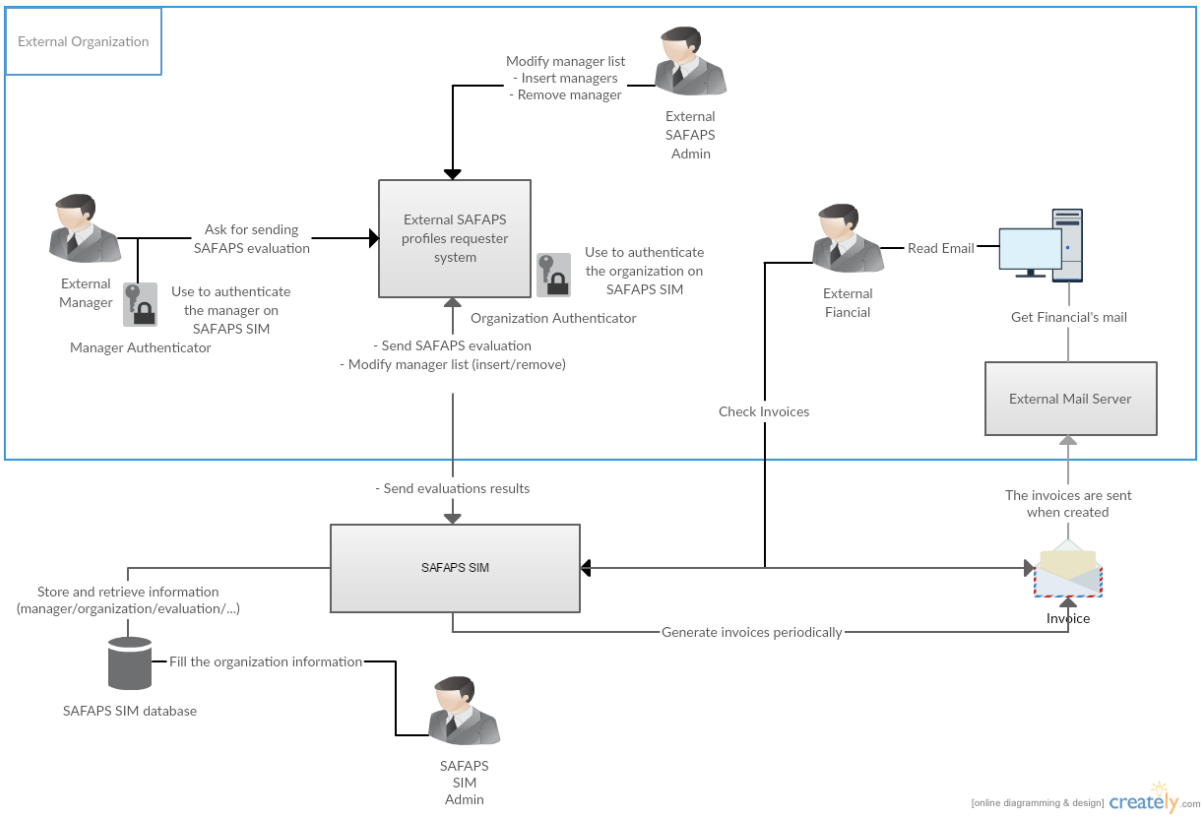


*Figure 2: Context diagram*

SAFAPS SIM is a system that is mainly used by external systems from organizations. Managers from such organizations can send SAFAPS through their external system to SAFAPS SIM and get authenticated using unique authenticating keys. Managers cannot interrogate SAFAPS SIM directly.

Once the evaluation result is ready, it is sent back to the external system.

When managers are added to or removed from the external system, it can notify SAFAPS SIM so that they are added to or removed from SAFAPS SIM.

External organizations' financials can consult invoices for the organization in two ways. He can either consult them from his mail or directly from SAFAPS SIM website. Indeed, when an invoice is generated by SAFAPS SIM, it is automatically sent to the external organization.

SAPAFS SIM is a simulation software. To limit the work to do on the back-end of SAFAPS website, the information about the organizations is filled in the system by the SAFAPS administrators.

## 3.2. Functional Viewpoint

## 3.3. Information Viewpoint

### 3.3.1. S&F Evaluation Request state

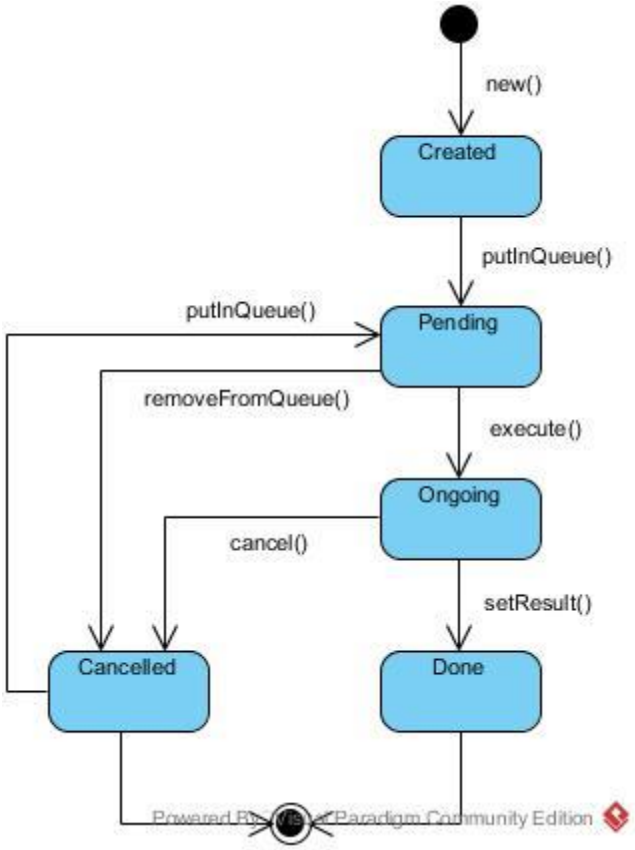


*Figure 3: State Chart of the S&F evaluation request*

The S&F evaluation request are to treated by the S&F evaluator asynchronously by the server. It means that the client is not blocked waiting for the result of the request. This asynchronous behaviour requires the S&F request to have a status stored in the database.

| State name | Description |
|---|---|
| Created | When the evaluation request is instantiated and inserted into the database, its status is created. |
| Pending | When the evaluation request is put in the queue awaiting to be treated by the S&F evaluator. |
| Ongoing | When the evaluation request is taken from the queue in order to be treated by the S&F evaluator |
| Done | When the result of the S&F evaluation request is set and inserted inside the database. |
| Cancelled | When the S&F evaluation request has been remove from the queue or the execution of the request has been interrupted. |

*Table 2: Description of S&F evaluation request states*

### 3.3.2. Database View

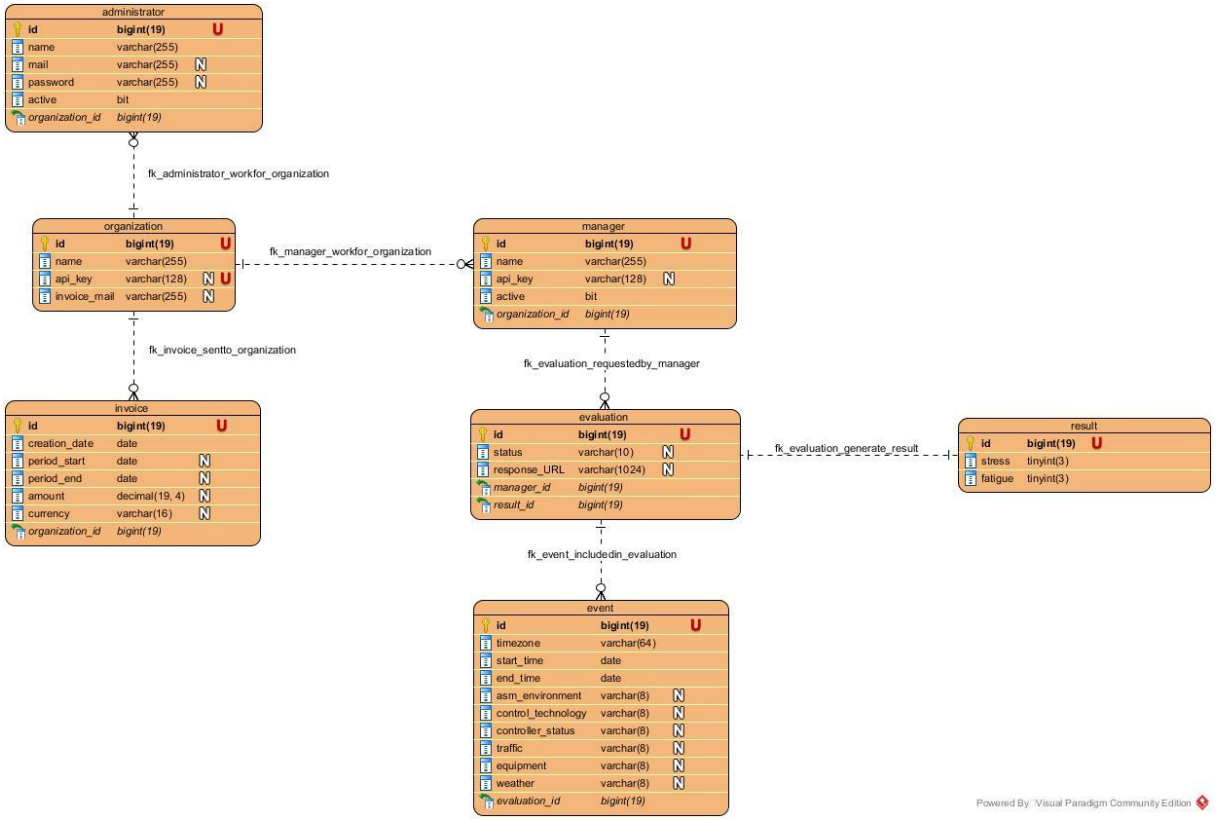


*Figure 4: Database entity diagram*

Some **additional information** regarding this diagram needs to be given in order to fully understand how to manipulated this presented data:

| Table | Column | Additional information |
|---|---|---|
| - administrators<br>- managers | - active | The active field represent whether the account is still authorized to use SAFAPS SIM functionalities. The type of this field represent a data with only 2 exclusive possible values. Depending on the database implementation, these values can either be TRUE/FALSE or 1/0. Both are correct. |
| - invoices | - currency | The currency of the invoice is stored as locale as describe in the RFC 4646 (e.g. en_US, en_UK). |
| | - amount | The amount is a floating number that can have up to 4 decimals. The stored value is the amount of the invoice converted into the currency stored in the invoice. |
| - evaluation | - status | It defines the status of the request. The possible values can be:<br>- "create"<br>- "pending"<br>- "ongoing"<br>- "done"<br>- "canceled" |
| - events | - timezone | The time zone of the event is stored as a string in the format "Continent/City". |
| | - start_time<br>- end_time | These "date" fields store a date in the calendar (MM/DD/YYYY) **and** a time (hh:mm:ss). On some database instance, this type is also called "datetime" |

*Table 3: Database entity additional information*

**Warning**: This diagram has been made without considering the database type or version which is used in the project. It offers a generic model showing how the data are stored and related to each other.

### 3.3.3. Database Standardization

Hereafter is the naming convention and other rules adopted for the database implementation.

- Tables:
  - The table names are fully given in low camel case (lowercase letter with '_' as space between words).

| Do | Don't |
|---|---|
| `organizations` | `Organizations` (first letter is uppercase) |
| `organization_invoices` | `organizationinvoices` (no '_' between words) |

*Table 4: Example of naming database tables*

- Table IDs:
  - Each table representing an entity must have an ID. Only tables made for many-to-many relationships may be without any ID.
  - Entities' ID are "bigint" stored over 19 bits
  - Entities' ID are **unique**, "**non-nullable"**, **primary keys**.
  - Entities' ID named 'id'
- Foreign keys:
  - Foreign keys are supposed to represent one-to-many or one-to-one relationships between two tables.
  - Each of these relationships needs to be defined using a verb.

Example:

| One… | Can be related to… | Verb |
|---|---|---|
| Organization | (many) Invoices | Invoices are => 'send to' => Organization |
| Evaluation | (many) Events | Events are => 'included in' => Evaluation |
| Evaluation | (one) Result | Result is => 'calculated from' => Evaluation |
| Result | (one) Evaluation | Evaluation is => 'generating' => Result |

*Table 5: Example of verbs for database table relationships*

**Note**: For one-to-one relationship, either one or the other table is able to store the foreign key (but not both at once).

  - Foreign keys must be named `<fk_table_name>_<fk_attribute_name>`

Example:

| Do | Don't |
|---|---|
| `organization_id` | `organizationid` |
| `evaluation_id` | `workfor` |
| `organization_id` | `administrator_organization_id` |

*Table 6: Example of naming foreign keys between database tables*

# 4. Quality Property Summary

# 5. Important Scenarios

# 6. Issues Awaiting Resolution

# 7. Appendices